title: Event Stream

import Tabs from '@theme/Tabs' import TabItem from '@theme/TabItem' import StreamEvent from '@site/docs/specs/mev-share/_streamEvent.mdx' import RemoteCodeBlock from "@site/src/components/RemoteCodeBlock"

Events on MEV-Share are distributed via an Server-Sent Events (SSE) streaming endpoint. Searchers listen to this endpoint to receive a stream of new events, which contain data they can use in their bundles. Currently, the events refer to Ethereum transactions.

Quickstart

Subscribe to the stream by making an HTTP GET request on the stream endpoint. Themev-share-client-ts library implements this as an asynchronous event handler.

```typescript import MevShareClient, { IPendingTransaction, IPendingBundle } from '@flashbots/mev-share-client'

const mevShareClient = MevShareClient.useEthereumMainnet(authSigner)

const txHandler = mevShareClient.on("transaction", async (tx: IPendingTransaction) => { / *Do something with the pending tx here.* / })

const bundleHandler = mevShareClient.on("bundle", async (tx: IPendingBundle) => { / *Do something with the pending bundle here.* / })

// call before your program terminates: txHandler.close() bundleHandler.close() ```

bash curl https://mev-share-goerli.flashbots.net

This will block until terminated manually (CTRL-C).

*Response:*

```bash :ping

data: {"hash":"0xc7dc06c994400830054ab815732d91275bc1241f9be62b62b687b7882f19b8d4","logs":null,"txs":
[{"to":"0x0000c335bc9d5d1af0402cad63fa7f258363d71a","functionSelector":"0x696d2073","callData":"0x696d20736861726969696969696e67"}]} ```

:::info Event Data

Events currently represent pending transactions, but eventually may be expanded to support other event types. For this reason we refer to this endpoint as an *event stream*, rather than a *transaction stream*.

:::

### Event Stream Endpoints

| Network | URL | |-|-| | Mainnet | https://mev-share.flashbots.net | | Goerli | https://mev-share-goerli.flashbots.net |

The endpoint sends an event with the message :ping every 15 seconds if no other messages were sent in the last 15 seconds.

# Event Scheme

Events dispatched via the SSE endpoint are JSON-encoded objects that adhere to the following scheme:

Note that each of these properties are optional; if a field is not present, it means that the transaction sender chose not to share that information.

Below is an example of a transaction event received from the stream:

json { "hash":"0xb756c9f6c34309d32c32daf1289c96d64a1068dfc2ead5e9bd0504640b91249e", "logs":[ { "address":"0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2", "topics":[
"0xddf252ad1be2c89b69c2b068fc378daa952ba7f163c4a11628f55a4df523b3ef", "0x0000000000000000000000001b59718eafa2bffe5318e07c1c3cb2edde354f9c",
"0x0000000000000000000000005c7bcd6e7de5423a257d81b442095a1a6ced35c5" ], "data":"0x000000000000000000000000000000000000000000000000161545fdcf782a85" }, { "address":"0xc02aaa39b223fe8d0a0e5c4f27ead9083c756cc2",
"topics":[ "0x7fcf532c15f0a6db0bd6d0e038bea71d30d808c7d98cb3bf7268a95bf5081b65", "0x0000000000000000000000005c7bcd6e7de5423a257d81b442095a1a6ced35c5" ],
"data":"0x000000000000000000000000000000000000000000000000161545fdcf782a85" }, { "address":"0x5c7bcd6e7de5423a257d81b442095a1a6ced35c5", "topics":[
"0x8ab9dc6c19fe88e69bc70221b339c84332752fdd49591b7c51e66bae3947b73c", "0x0000000000000000000000000000000000000000000000000000000000000089",
"0x0000000000000000000000000000000000000000000000000000000000117363", "0x0000000000000000000000003a23f943181408eac424116af7b7790c94cb97a5" ],
"data":"0x000000000000000000000000000000000000000000000000161fb772f99eaf7a000000000000000000000000000000000000000000000000161fb772f99eaf7a000000000000000000000000000000000000000000000000161fb772f99eaf7a00
} ], "txs":[ { "to":"0x5c7bcd6e7de5423a257d81b442095a1a6ced35c5", "functionSelector":"0x44b8be68",
"callData":"0x44b8be680000000000000000000000003a23f943181408eac424116af7b7790c94cb97a5000000000000000000000000b658ba58f7153e99c05c9b7610f17bfeeab6bff5000000000000000000000000c02aaa39b223fe8d0a0e5c4f27ead908
} ] }

### Understanding double-hash

Note that the hash field is actually a keccak256 hash of the underlying bundle/transaction hash, essentially a double-hash.

Below is code-snippet in golang to calculate double-hash for testing purposes.

```go package main

import ( "fmt"

"github.com/ethereum/go-ethereum/common"
"golang.org/x/crypto/sha3"

)

func main() { underlyingHash := common.HexToHash("0xd2d662b8aa0e8d86ea75d363522c9ede42ef538ae353da564d501c044a885293") doubleHasher := sha3.NewLegacyKeccak256() doubleHasher.Write(underlyingHash.Bytes()) dHash := doubleHasher.Sum(nil) matchingHash := common.BytesToHash(dHash) fmt.Println(matchingHash.String()) //prints 0x90b4f5664cc201c3aa112d6bb2fa414c4aee10f00994692b282c1d14a1db6e4d } ```

Now that you've started listening to transactions, you're almost ready to start searching! Read on to the next page to learnabout bundles.

# Historical Data

Historical hints can be retrieved from the historical hint API supported by the event stream endpoint. Each hint is associated with a block number and timestamp. Block number is the latest Ethereum block number at the time the hint was generated. Timestamp is the timestamp at the time the hint was generated.

### GET /api/v1/history/info

Returns information about the available historical hint data.

#### Response

| Field | Type | Description | |-|-|-| | count | number | The number of historical hints available. | | minBlock | number | The earliest block number for which historical hints are available. | | maxBlock | number | The latest block number for which historical hints are available. | | minTimestamp | number | The earliest timestamp for which historical hints are available. | | maxTimestamp | number | The latest timestamp for which historical hints are available. | | maxLimit | number | The maximum number of historical hints that can be requested in a single request. |

### GET /api/v1/history

#### Query Parameters

| Field | Type | Description | |-|-|-| | blockStart (optional) | number | The block number to start retrieving historical hints from. | | blockEnd (optional) | number | The block number to end retrieving historical hints from. | | timestampStart (optional) | number | The timestamp to start retrieving historical hints from. | | timestampEnd (optional) | number | The timestamp to end retrieving historical hints from. | | limit (optional) | number | The maximum number of historical hints to retrieve. Default limit is maxLimit. | | offset (optional) | number | The offset to start retrieving historical hints from. |
```

Response

Returns an array of historical hints.

| Field | Type | Description | |-|-|-| | block | number | The block number associated with the historical hint. | | timestamp | number | The timestamp associated with the historical hint. | | hint | Hint | Hint as it was sent to the live streaming endpoint in the past. |

Example

Get information about historical hint data

```
bash curl https://mev-share-goerli.flashbots.net/api/v1/history/info
```

Response:

json { "count": 20146, "minBlock": 9091377, "maxBlock": 9143624, "minTimestamp": 1685452445, "maxTimestamp": 1686225251, "maxLimit": 500 }

Get historical event data beginning at start of stream history

```
bash curl https://mev-share-goerli.flashbots.net/api/v1/history
```

Get historical hint data from a specific block range

```
bash curl 'https://mev-share-goerli.flashbots.net/api/v1/history?blockStart=9091377&blockEnd=9091379'
```

Response:

json [{ "block": 9091377, "timestamp": 1685452445, "hint": { "txs": [{ "to": "0x8d460b72eaf3d63830e16c22d1fc6908d0834abe", "callData": "0x", "functionSelector": "0x00000000" }], "hash": "0x50df4922dd5f9adee91d44119132da85b50fe61f0c77556b039261f7828e1794", "logs": null, "gasUsed": "0x5208", "mevGasPrice": "0x3b9aca00" } }, { "block": 9091379, "timestamp": 1685452489, "hint": { "txs": null, "hash": "0x40a85a6e37b449033924da72c0cf9dabcf2ac726b5a88f0ceff330f11bd01913", "logs": null, "gasUsed": "0xaae60", "mevGasPrice": "0x45a9b5b00" } }]

Querying with Offset & Limit

Event history results are returned in chunks whose size are defined bylimit, the maximum limit being specified in the/history/info endpoint.

```bash

# assuming the limit is 500

curl 'https://mev-share-goerli.flashbots.net/api/v1/history?blockStart=9091377' curl 'https://mev-share-goerli.flashbots.net/api/v1/history?blockStart=9091377&offset=500' curl 'https://mev-share-goerli.flashbots.net/api/v1/history?blockStart=9091377&offset=1000' curl 'https://mev-share-goerli.flashbots.net/api/v1/history?blockStart=9091377&offset=1500'

# or with a custom limit

curl 'https://mev-share-goerli.flashbots.net/api/v1/history?blockStart=9091377&limit=100' curl 'https://mev-share-goerli.flashbots.net/api/v1/history?blockStart=9091377&limit=100&offset=100' curl 'https://mev-share-goerli.flashbots.net/api/v1/history?blockStart=9091377&limit=100&offset=200' curl 'https://mev-share-goerli.flashbots.net/api/v1/history?blockStart=9091377&limit=100&offset=300' ```
```